User Manual

for

AD2100-12-1M-104P

Dual Channel, 100 MSPS, 12-bit, PC/104-PLUS Analog Input Module

CHASE SCIENTIFIC COMPANY
P.O. Box 1487
Langley, WA 98260

Tel: 360-221-8455
Fax: 360-221-8457

Email: techsupport@chase2000.com
Web: http://www.chase2000.com

Original Document: AD2100-12-1M-104P_manual.odt

(created 05-06-2004, updated *6/27/2007*)

© Copyright 2004, 2005 by
Chase Scientific Company

This manual, the AD2100 module, and the software drivers outlined in this document are copyrighted with all rights reserved. Under the copyright laws, the above mentioned may not be copied, in whole or in part, without the express written consent of Chase Scientific Company.

TABLE OF CONTENTS

1 GENERAL INFORMATION....4

1.1 Introduction....4
1.2 References....4
1.3 Deliverables....4
1.3.1 Software....4
1.3.2 Hardware....5
1.3.3 Checklist....5
1.4 Product Specification....5
1.5 Technical Support / Software Updates....6
1.6 Warranty....6

2 HARDWARE DESCRIPTION....8

2.1 Introduction....8
2.2 Block Diagram....8
2.3 Board Drawing....9
2.4 Board I/O....10
2.4.1 Header Pinouts and Pin Descriptions....10
2.5 Hardware Register Definitions....11
2.6 Analog Input Ranges and Resolution....11

3 THEORY OF OPERATION....12

3.1 Introduction....12
3.2 Acquiring Analog Input Data....12

4SOFTWARE DRIVERS....12

4.1 Introduction....12
4.2 Driver Installation....12
4.2.1 Windows 2000/XP....12
4.2.2 Linux Installation (Kernels 2.0x, 2.2x, 2.4x, 2.6x)....13
4.2.3Windows 98/ME/NT4....14
4.3 Function Calls....14
4.3.1 Function Declarations in C....14
4.3.2 Function Call Descriptions / Usage....15
4.3.2.1 ad2100_CountCards()....15
4.3.2.2 ad2100_Open()....16
4.3.2.3 da2100_Close()....16
4.3.2.4 ad2100_Reset()....17
4.3.2.5 ad2100_SetClockRate()....17
4.3.2.6 ad2100_SetSegment_Size()....18
4.3.2.7 ad2100_SetNumSegments()....18
4.3.2.8 ad2100_SetTrigPol()....19
4.3.2.9 ad2100_SoftTrig()....19
4.3.2.10 ad2100_Digitize()....19
4.3.2.11 ad2100_Digitize_Done()....20
4.3.2.12 ad2100_TransferData()....20
4.4Programming Examples....21
4.4.1 Windows/Linux Example Using Generic C/C++....21

5MISCELLANEOUS....22

5.1Calibration....22
5.2Maintenance....23

ILLUSTRATIONS / TABLES

FIGURE 1 – BLOCK DIAGRAM..... 8

FIGURE 2 – BOARD LAYOUT..... 9

FIGURE 3 - 4-PIN DIGITAL I/O CONNECTOR..... 10

1 GENERAL INFORMATION

1.1 Introduction

The AD2100-12-1M-104P is a PC/104-Plus analog I/O module which features two (2) 12-bit, high speed singled-ended (50 ohms) synchronous input channels, each capable of capturing data at 100 MSPS with a minimum bandwidth of 50 MHz. The AD2100 module also comes configured with (4) General Purpose I/O and operates over industrial temperature range standard (40°C to 85°C).

Analog inputs, trigger input, and clock inputs are available on SMA connectors. Each high speed channel has 1Meg memory per channel of software selectable onboard memory using programmable segment sizes. Each segment is independently triggered and packed into memory. Trigger source can be either an external TTL trigger or on-board software trigger.

Timing is controlled by an onboard clock circuit referenced to a 100 MHz crystal oscillator and can be programmed for 100MHz, 50MHz, ..., 6.25MHz. Once A/D conversions are completed, data can be transferred from the module to main memory via direct memory reads using simple function calls.

1.2 References

PC/104-Plus Specification, Version 2.0, November 2003 by PC/104 Embedded Consortium.

"P996.1 Standard for Compact Embedded-PC Modules"; PC/104 Specification, Version 2.3, June 1996 by PC/104 Consortium.

"IEEE P996 draft standard", 1987 by IEEE Standards Committee. This bus standard was the basis for the first PC/104 standard released in 1992.

1.3 Deliverables

1.3.1 Software

The AD2100 comes with drivers for **Windows 98/NT/2000/XP**. Software comes on a single 3.5" diskette. Call Chase Scientific for the latest information on drivers for other operating system platforms or check the web site for the latest updates.

Software drivers are provided as a Dynamic Link Library (*.DLL) which is compatible with most 32-bit windows based development software including Microsoft C/C++, Borland C/C++, and Borland Delphi. This DLL uses the "**cdecl"** calling convention which is default capatible with the compilers above and provides easy to use function calls to the system drivers "windrvr6.vxd" for Windows 98/NT and "windrvr6.sys" for Windows 2000/XP.

Actual Listing of files on Diskette/CD for Windows:

```
-----------   D I R E C T O R I E S / F I L E S  ----------------
BASE_DIR
|
| readme.txt                 // This file.
|
| Register_AD2100_Win2000_XP.bat     // Installs Kernel driver for Win2000/XP
| UnRegister_AD2100_Win2000_XP.bat   // Uninstalls Kernel driver for Win2000/XP
```

```
|
| Register_AD2100_Win98_ME_NT4.bat   // Installs Kernel driver for Win98/ME/NT4.0
| UnRegister_AD2100_Win98_ME_NT4.bat // Uninstalls Kernel driver for Win98/ME/NT4.0
|
| wdreg16.exe        // Called by Register_AD2100_Win98_ME_NT4.bat
| wdreg.exe          // Called by Register_AD2100_Win2000_XP.bat
| windrvr6.inf       // Setup information file automatically called by above exe(s).
|
| a2100_dll.dll          // DLL for 98/ME/NT4/2000/XP  ( extern "C"
__declspec(dllimport) )
| a2100_dll_import.h     // Header file for DLL
| a2100_dll.lib          // Library file for DLL in Borland C++
|
| | Microsoft VC++ DLL Files
| | | awg1200_dll_msvc.lib     // Include in MSVC Project to compile DLL above
| | | awg1200_dll_import.h     // Header file for DLL
|
|
| ad2100_diag_project.exe           // Simple GUI to test DLL and Kernel drivers
|
| AD2100_PC104P.inf   // Plug-And-Play file needed by 98/ME/NT4/2000/XP for automatic
|                      // hardware configuration.
|
| windrvr6.sys    // Windows 98/ME/NT4/2000/XP Driver - copy this virtual driver
|                 // to "c:\<windir>\system32\drivers\" if not automatically done
|                 // so after running batch file.
|

--------------   E N D    ----------------
```

1.3.2 Hardware

The AD2100 hardware consists of a single PC/104-Plus compliant module. The module is shipped with a manual with complete hardware and software descriptions. The ISA bus connector (PC/104 only portion) is configured as a passthrough mode but does not directly connect to any circuitry on the card. This card can only be accessed as a PCI device according to PC/104-Plus specification.

1.3.3 Checklist

Item #	Qty	Part Number	Description
1	1	AD2100-12-1M-104P	Dual 100 MSPS, 12-bit, A/D PC/104-Plus Module
2	1	AD2100-Drivers	3.5” diskette with Dynamic Link Libraries for Windows 98/NT/2000/XP. Includes "example_snippet.txt".
3	1	AD2100-Manual	User manual for AD2100 board and software drivers.

1.4 Product Specification

(all specifications are at 25C unless otherwise specified)

HIGH SPEED ANALOG INPUTS (CH1-2)

Number of inputs	(2) Synchronous, Single Ended into 50 ohms
A/D Resolution	12-bit (1 part in 4096)
Input Bandwidth	50 MHz minimum @ 5V pk to pk (Av=1)
Input Range	The A/D count of 0 equals -0.525V. A count of 4095 equals +0.525V. (order option 3 for user specified FIXED gain)
Acquisition Time to Full-Scale Step	16 ns typical to 0.01% (1/2 LSB)
Input Coupling	50 ohms load to ground/shield

Overvoltage Protection	+/- 5V Maximum
Nonlinearity	+/- 2 LSB typical
Conversion Rate / channel	100 MHz maximum, Programmable down to 6.25MHz (x2 increments)
Memory Size	1048576 samples maximum
Segment Sizes	Software selectable segment sizes of 128, 256, 512, 1024, 2048, 4096, and 8192. Maximum number of segments that can be stored in memory is 1024
Acquisition Mode	Single-shot initiated by software, external or internal triggering (see trigger sources)
Timebase	100 MHz, 50 MHz, 25 MHz, 12.5 MHz, 6.25MHz (Software Selectable)
Trigger	
Location	Fixed trigger position at beginning of each segment
Sources	external TTL signal, internal software trigger
Slope	Positive, Negative
Coupling	DC
External Clock Input	
Frequenc Range	1 MHz through 100 MHz
Amplitude Range	0dBm - dBm (square/sine wave)
Coupling	AC

GENERAL

Power Supply (Vcc)	+5V @ 860mA (6.25 MSPS) +5V @ 1240mA (100 MSPS)
Operating Temperature	0 to 70 degrees C standard -40 to +85 degrees C extended
Operating Humidity	5 to 95% non-condensing
Size	PC/104-Plus: 3.55" x 3.775"
Data Bus	32-bits PCI

1.5 Technical Support / Software Updates

For technical support:

Phone	(360) 221-8455
Fax	(360) 221-8457
Email	techsupport@chase2000.com
Mail	Chase Scientific Company P.O. Box 1487 Langley, WA 98260

For software updates:

Email	techsupport@chase2000.com
Web	http://www.chase2000.com

1.6 Warranty

Chase Scientific Company (hereafter called Chase Scientific) warrants to the original purchaser that its AD2100-12-1M-104P, and the component parts thereof, will be free from defects in workmanship and materials for a period of ONE YEAR from the data of purchase.

Chase Scientific will, without charge, repair or replace at its option, defective or component parts upon delivery to Chase Scientific's service department within the warranty period accompanied by proof of purchase date in the form of a sales receipt.

EXCLUSIONS: This warranty does not apply in the event of misuse or abuse of the product or as a result of unauthorized alterations or repairs. It is void if the serial number is altered, defaced or removed.

Chase Scientific shall not be liable for any consequential damages, including without limitation damages resulting from loss of use. Some states do not allow limitation or incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific rights. You may also have other rights that vary from state to state.

Chase Scientific warrants products sold only in the USA and Canada. In countries other than the USA, each distributor warrants the Chase Scientific products that it sells.

NOTICE: Chase Scientific reserves the right to make changes and/or improvements in the product(s) described in this manual at any time without notice.

2 HARDWARE DESCRIPTION

2.1 Introduction

The AD2100 hardware consists of the following major functions:

- (2) 12-bit, 100 MSPS synchronous, single-ended, bipolar A/D Inputs (DC coupled)
- (1) TTL Trigger Input
- (1) TTL, ECL, Sinewave Clock Input (AC coupled)

2.2 Block Diagram

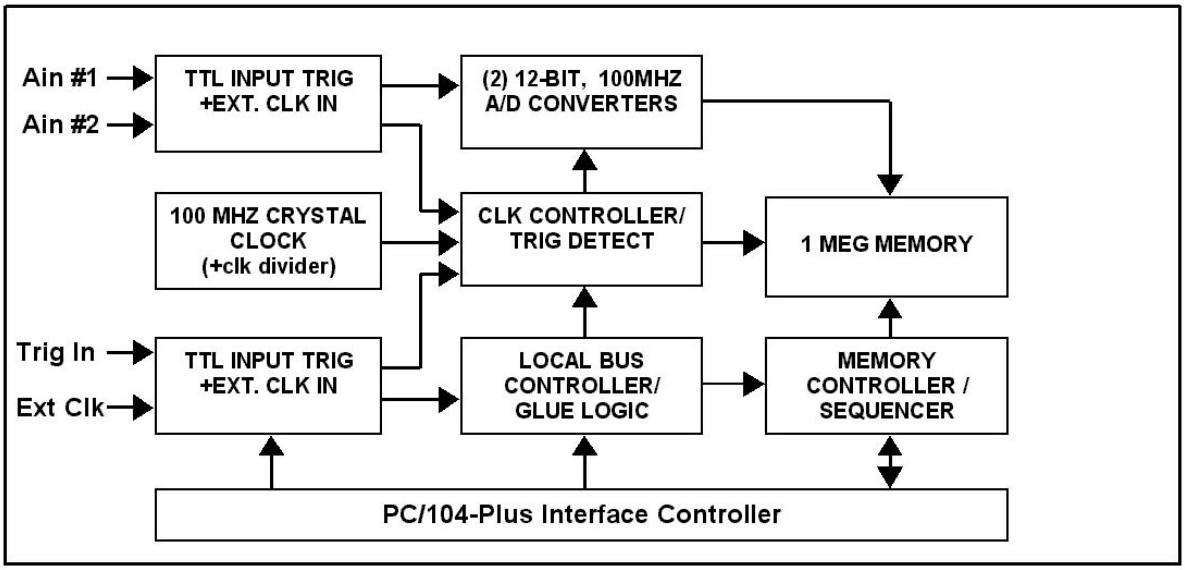


Figure 1 – Block Diagram

2.3 Board Drawing

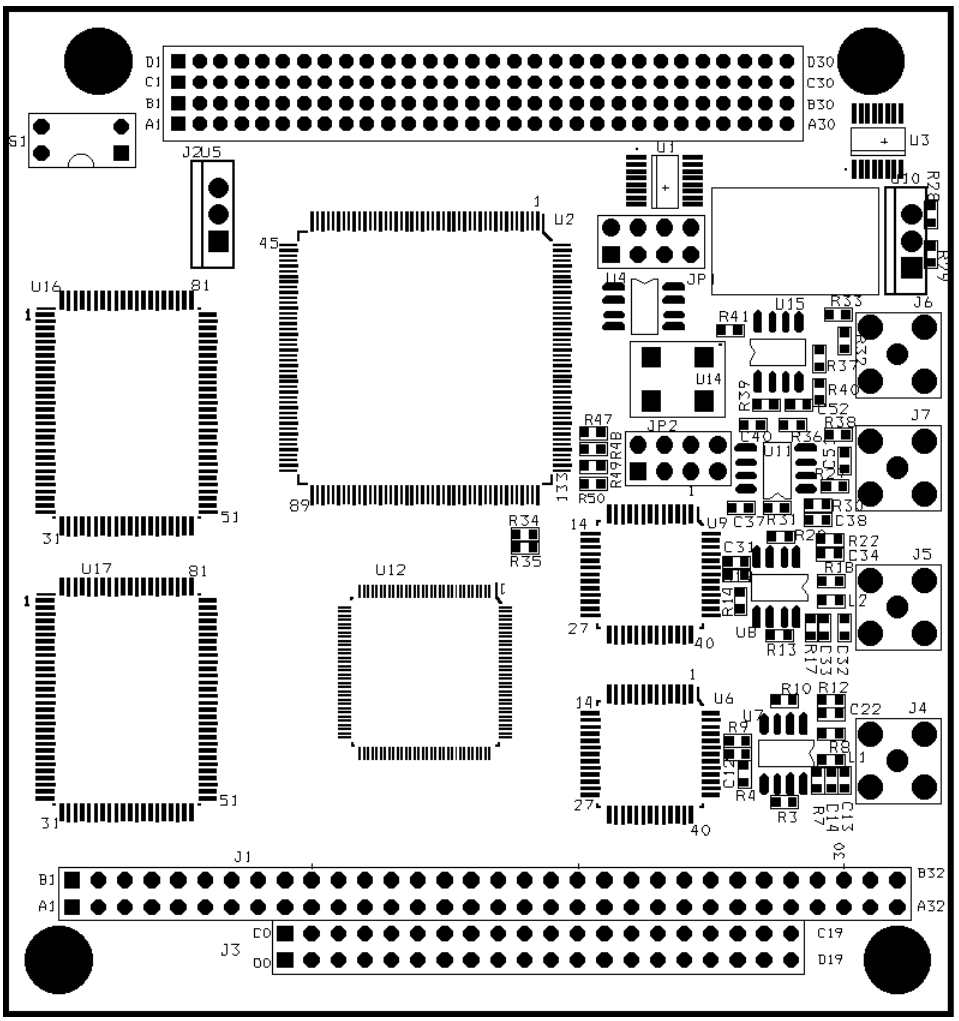


Figure 2 – Board Layout

2.4 Board I/O

2.4.1 Header Pinouts and Pin Descriptions

4-Pin General Purpose Digital I/O

I/O CONNECTOR JP2 (standard 100 mil spacing)

TOP VIEW

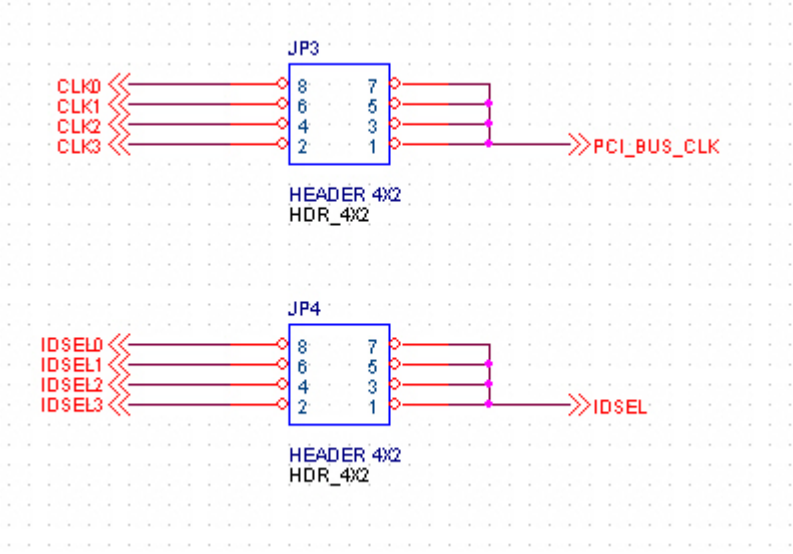
GND	1	2	GPIO_1
GND	3	4	GPIO_2
GND	5	6	GPIO_3
GND	7	8	GPIO_4

Figure 3 - 4-Pin Digital I/O Connector

Signal Name	Definition
GPIO_1	Digital input/output bit 1
GPIO_2	Digital input/output bit 2
GPIO_3	Digital input/output bit 3
GPIO_4	Digital input/output bit 4

See software driver section for explanation of how to program these bits.

IDSEL and BUS CLOCK SELECT Headers ==>



2.5 Hardware Register Definitions

Due to hardware protocol complexity, the details on how to use these ports are beyond the scope of this manual. The software drivers should provide an effortless integration path for the user. However, if the need arises that is beyond what the software drivers can provide, please call the factory for technical support.

2.6 Analog Input Ranges and Resolution

Analog input range is by default +/- 0.528V. Consult factory for user gain setting options. The resolution is 12 bits.

3 THEORY OF OPERATION

3.1 Introduction

Because the PC/104-Plus bus is slower than the digitizing rate of the AD2100, the data must first be digitized and stored on the AD2100 and then transferred to system memory at the slower bus speed.

3.2 Acquiring Analog Input Data

Before capturing data, the user must first setup the correct digitizing rate, trigger input mode, trigger polarity, number of segments, segment size, and pretrigger value (if applicable).

After proper setup is performed, you're ready to digitize. When the digitize function is called, the A/D converter is activated and begins storing data into buffer RAM. If multiple segmented memory is used where triggered blocks of memory is packed together until some set memory limit is met, then the USER must set up the segment size and number of segments to be digitized.

If post/pre-triggering is enabled, then the trigger detect circuit will initially be suppressed until the user programmed pretrigger time has elapsed. Then, after the pretrigger timer has completed and the trigger event has occurred, the A/D data will continue to be stored into the onboard buffer RAM until the postrigger timer has run out.

4 SOFTWARE DRIVERS

4.1 Introduction

The primary objective in designing software drivers is to get the user up and running as quickly as possible. The details on installing software drivers are listed in 4.2.xx. While the listing of function calls and their parameter definitions are listed in section 4.3.xx, the programming examples in section 4.4.x will show you how to include them into your programs.

The drivers are designed to work under Windows 98/ME/NT4/2000/XP.

4.2 Driver Installation

4.2.1 Windows 2000/XP

1) Do not install AD2100 card at this time.

2) UnZip all files into directory "C:\temp\AD2100\" (create directories if needed) You can move and/or copy the files later to a directory of your choice.

3) Run Register_AD2100_Win2000_XP.bat. This will copy the Kernel driver windrvr6.sys to "c:\<windir>\system32\drivers\" directory and will register the Kernel driver in the Windows Registry so

that it starts up each time the computer is rebooted.

4) Power off computer. Insert AD2100 card. Power up computer.

5) When OS asks for Driver File point to "AD2100_PC104P.inf". If OS does not ask for file, then check hardware configuration and update if not listed properly under "Jungo" in Device Manager (see below).

To check to see which driver is installed, do the following:

```
=> Control Panel
  => System
    => Hardware
      => Device Manager
        => Jungo
            AD2100_PC104P
            WinDriver
```

If you see another driver in place of "AD2100_PC104P", then right click the first device under Jungo and click properties. Update the driver by pointing to "AD2100_PC104P.inf". You may have to go through a series of menus.

4.2.2 Linux Installation (Kernels 2.0x, 2.2x, 2.4x, 2.6x)

Example Linux Installation on RH9.0 (version 2.4.20-8) 02/04/2005
---

1. Before you start

Install Red Hat Linux 9.0 (Kernel 2.4.20-8). Make sure the GNU C++ compiler, tools, and libraries are installed. This requires using the custom install mode or simply installing all packages.

2. Install the Kernel Driver

Make sure you log on as administrator (root level).

Copy the archive file "ad2100.tgz" from the CD (or download) to the directory "/root/ad2100" (for example). Open the ad2100.tgz by using the archive manager "File Roller" or use "tar xvzf ad2100.tgz". This will create two folders, "WinDriver" and "ad2100_src" in this directory.

Copy the "WinDriver" folder to the directory "/usr/local". This will create the directory "/usr/local/WinDriver" with all the support files needed to recompile the WinDriver Kernel.

Type the following commands in a terminal windows while in directory "/usr/local/WinDriver/redist":

A) # ./configure
NOTE: The configure script creates a makefile based on your specific running kernel.

B) # make

C) # make install

3. Install the ad2100 Interface Driver with Sample (user) Test Program.

To compile sample test program change to directory "/root/ad2100/ad2100_src". Then type "make all". This will compile, link, and create the interface library ad2100_o.o as well as the test application, ad2100_test.

Just use the test application as a skeleton for your own interface/application program.

4.2.3 Windows 98/ME/NT4

1) Do not install AD2100 card at this time.

2) UnZip all files into directory "C:\temp\AD2100\" (create directories if needed) You can move and/or copy the files later to a directory of your choice.

3) Run "Register_AD2100_Win98_ME_NT4.bat". This will copy the Kernel driver windrvr6.sys to "c:\<windir>\VMM32\" directory and will register the Kernel driver in the Windows Registry so that it starts up each time the computer is rebooted.

4) Power off computer. Insert AD2100 card. Power up computer.

5) When OS asks for Driver File point to "AD2100_PC104P.inf". If OS does not ask for file, then check hardware configuration and update if not listed properly under "Jungo" in Device Manager (see below).

To check to see which driver is installed, do the following:

=> Control Panel
=> System
=> Hardware
=> Device Manager
=> Jungo
AD2100_PC104P
WinDriver

If you see another driver in place of "AD2100_PC104P", then right click the first device under Jungo and click properties. Update the driver by pointing to "AD2100_PC104P.inf". You may have to go through a series of menus.

4.3 Function Calls

4.3.1 Function Declarations in C

Header File for C Function Calls =>

```
//---------------------------------------------------------------------------
#ifndef ad2100_dllH
```

```
#define ad2100_dllH
//---------------------------------------------------------------------------

#define IMPORT extern "C" __declspec(dllimport)

DWORD TempArray[2097152];
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <io.h>
#include <string.h>


// User Function calls
IMPORT DWORD ad2100_CountCards(void);
IMPORT DWORD ad2100_Open(DWORD CardNum);
IMPORT DWORD ad2100_Close(DWORD CardNum);

IMPORT void ad2100_Reset(DWORD BrdNum);
IMPORT void ad2100_SetClockRate(DWORD BrdNum, DWORD DivideBy);

IMPORT void ad2100_SetSegment_Size(DWORD BrdNum, DWORD SegSize);
                                    // 0,1,2,3,4,5,6=>
                                    //128,256,512,1024,2048,4096,8192
IMPORT void ad2100_SetNumSegments(DWORD BrdNum, DWORD NumSeg);
                                    // NUM_SEGS*SEG_SIZE < 2MEG (2097152)
IMPORT void ad2100_SetTrigPol(DWORD BrdNum, DWORD TrigPol);
                                    // 0=Pos Edge, 1=Neg Edge
IMPORT void ad2100_SoftTrig(DWORD BrdNum);

IMPORT void ad2100_Digitize(DWORD BrdNum);
IMPORT DWORD ad2100_Digitize_Done(DWORD BrdNum);
IMPORT void ad2100_TransferData(DWORD BrdNum,
                                DWORD SEG_SIZE,
                                DWORD NUM_SEGS,
                                PVOID UserArrayPtr);

#endif
```

4.3.2 Function Call Descriptions / Usage

4.3.2.1 ad2100_CountCards()

Description
Returns number of AD2100 cards present on computer.

Declaration
```
DWORD ad2100_CountCards(void);
```

Parameters
none

Return Value
Returns with an encoded value which represents the number of AD2100.

Return Values:

0: Kernel Driver ok, but no card found
1-4: Normal range of possible number of cards detected
13: Kernel Driver not working

Example
```
DWORD Num_ad2100_Boards = ad2100_CountCards();
```

4.3.2.2 ad2100_Open()

Description
Loads the AD2100 software drivers and sets the AD2100 board to its default state.

Declaration
```
DWORD ad2100_Open(DWORD CardNum);
```

Parameters
CardNum: 1 <= CardNum <= 4

Return Value
Returns with error code. A "0" means everything is fine. See below for details for other values.

Return Values:
0: Opened Windriver Successfully and AD2100 Card Found Successfully
1: Opened Windriver Successfully, but NO AD2100 CARDS FOUND
2: Opened Windriver Successfully, AD2100 card found, but Cannot Open Driver to other Board Functions.
3: Opened Windriver Successfully, Board already open.
6: Card number exceeds number of cards.
13: FAILED TO OPEN Windriver Kernel Driver

Example
```
DWORD OpenErrorCode = ad2100_Open(1);  // Opens Board Number 1 and stores value.
```

4.3.2.3 da2100_Close()

Description
Closes AD2100 drivers. Should be called after finishing using the driver. However, if no other software uses the “windrv.xxx” (usual situation), then there is no need to close it until user is ready to completely exit from using their main software program.

Declaration
```
DWORD ad2100_Close(DWORD CardNum);
```

Parameters
CardNum: 1 <= CardNum <= 4

Return Value
Returns with error code. A "0" means everything is fine. See below for details for other values.

Return Values:
0: Closed Windriver Successfully for AD2100 card requested.
5: AD2100 Card Already Closed for card requested.
13: FAILED TO ACCESS Windriver Kernel Driver

Example
```
DWORD CloseErrorCode = ad2100_Close(1);
```

4.3.2.4 ad2100_Reset()

Description
Resets memory counters and triggering functions (re-arms). This should be called immediately after opening card for the first time and every time before "ad2100_Digitize" is called .

Declaration
```
void ad2100_Reset(DWORD BrdNum);
```

Parameters
CardNum: 1 <= CardNum <= 4

Return Value
None.

Example
```
ad2100_Reset1);
```

4.3.2.5 ad2100_SetClockRate()

Description
Sets internal clock rate based on on-board clock or external clock (DivideBy=8).

Declaration
```
void ad2100_SetClockRate(DWORD BrdNum, DWORD DivideBy);
```

Parameters
CardNum: 1 <= CardNum <= 4
DivideBy:
- 0: 100 MegaSamples/second
- 1: 50 MegaSamples/second
- 2: 25 MegaSamples/second
- 3: 12.5 MegaSamples/second
- 4: 6.25 MegaSamples/second

- 8: External Clock Select
  Frequency Range: 1MHz - 100 MHz
  Amplitude: 0dBm - 6dBm

Return Value
None.

Example
```
ad2100_SetClockRate(1,0); // Set board # 1 to 100 MegaSamples
```

4.3.2.6 ad2100_SetSegment_Size()

Description
Sets the segment size. If NumSegments (see "ad2100_SetNumSegments") is set to 1, you can use all segment sizes listed below. If using multiple segments (more than 1 segment), then you can only use SegSize: 0 <= SegSize <= 5.

Declaration
```
void ad2100_SetSegment_Size(DWORD BrdNum, DWORD SegSize);
```

Parameters
BrdNum: 1 <= CardNum <= 4

SegSize: Value **Description**

Value	Description
0	128 Samples / Segment
1	256 Samples / Segment
2	512 Samples / Segment
3	1024 Samples / Segment
4	2048 Samples / Segment
5	4096 Samples / Segment
6	8192 Samples / Single Segment Only
7	16384 Samples / Single Segment Only
8	32768 Samples / Single Segment Only
9	65536 Samples / Single Segment Only
10	131072 Samples / Single Segment Only
11	262144 Samples / Single Segment Only
12	524288 Samples / Single Segment Only
13	1048576 Samples / Single Segment Only
14	2097152 Samples / Single Segment Only

Return Value
None.

Example
```
ad2100_SetSegment_Size(1,3); // Set board #1 for Segment Size = 1024 Samples
```

4.3.2.7 ad2100_SetNumSegments()

Description
Sets number of segments before digitize is complete. User can also call reset at any time to start over if number of triggers is not sufficient to finish the digitize process.

Declaration
```
void ad2100_SetNumSegments(DWORD BrdNum, DWORD NumSeg);
```

Parameters
BrdNum: 1 <= CardNum <= 4
NumSeg: (NumSeg*SegSize) <= 2MEG [2097152]

Return Value
None.

Example
```
ad2100_SetNumSegments(1,3); // Set board #1 for 3 Segments
```

4.3.2.8 ad2100_SetTrigPol()

Description
Sets which edge will cause a trigger to occur.

Declaration
```
void ad2100_SetTrigPol(DWORD BrdNum, DWORD TrigPol);
```

Parameters
BrdNum: 1 <= CardNum <= 4
TrigPol:
0: Positive Edge Triggered
1: Negative Edge Triggered

Return Value
None.

Example
```
ad2100_SetTrigPol(1,0); // Set board #1 for Positive Edge Trigger
```

4.3.2.9 ad2100_SoftTrig()

Description
Creates a trigger as if an external hardware trigger action occurred.

Declaration
```
void ad2100_SoftTrig(DWORD BrdNum);
```

Parameters
BrdNum: 1 <= CardNum <= 4

Return Value
None.

Example
```
ad2100_SoftTrig(1); // Creates trigger under software control for Brd #1
```

4.3.2.10 ad2100_Digitize()

Description
Must always call "ad2100_Reset()" first to reset memory controller. Board must be open, of course (see ad2100_Open).

All other functions such as SetClockRate, SetSegmentSize, SetNumSegments, SetTrigPol do not have to be called again unless changes are made. Reset is the only function that is required to called each time before a new digitize.

Declaration
```
void ad2100_Digitize(DWORD BrdNum);
```

Parameters
BrdNum: 1 <= CardNum <= 4

Return Value
None.

Example
```
ad2100_Digitize(1); // Starts to digitize on board #1
```

4.3.2.11 ad2100_Digitize_Done()

Description
This function is used to check to see if "ad2100_Digitize()" had comleted. This call can be put into a multiplexed timing loop (or thread) to avoid tying up the microprocessor.

When this function comes back "1", then the user can transfer the data safely.

Declaration
```
DWORD ad2100_Digitize_Done(DWORD BrdNum);
```

Parameters
BrdNum: 1 <= CardNum <= 4.

Return Value
0: False
1: True

Example
```
DWORD BoardDone = ad2100_Digitize_Done(1);
              // Check to see if Board #1 has completed the digitize process.
```

4.3.2.12 ad2100_TransferData()

Description
Transfers the stored digitized data located on the PC/104 module to the System memory location pointed to by "UserArrayPtr". "ad2100_Digitize_Done" much be True before calling this function.

Declaration
```
void ad2100_TransferData(DWORD BrdNum, DWORD SEG_SIZE, DWORD NUM_SEGS, PVOID
UserArrayPtr);
```

Parameters
BrdNum: 1 <= CardNum <= 4
SEG_SIZE: 128 <= SEG_SIZE <= Memory Size in powers of 2.
NUM_SEGS: (NUM_SEGS*SEG_SIZE) <= 2MEG

UserArrayPtr: Pointer to User's Array.

Return Value
None.

Example
```
ad2100_TransferData(1,128,1,UserArrayPtr); // Transfers 1 segment worth of 128
       samples data from board #1 to users array pointed to by UserArrayPtr.
```

4.4 Programming Examples

4.4.1 Windows/Linux Example Using Generic C/C++

```
//////////////////////////////////////////////////////////////
// File - ad2100_test.c
//
// Example code for accessing the AD2100 PC/104+ module.
//
//////////////////////////////////////////////////////////////


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "ad2100_o.h"


int main(int argc, char *argv[])
{

//--------------------------------------------------------------------
// 1) INITIALIZE VARIABLES
//--------------------------------------------------------------------

    DWORD TempArray[8192];                                  printf("\n");
    DWORD NumCards;
    DWORD ad2100_error_code;

    for (DWORD x=0; x<8192; x++) TempArray[x] = 0;

//--------------------------------------------------------------------
// 2) CHECK FOR NUMBER OF AD2100 CARDS, OPEN DRIVERS, REPORT ANY ERRORS
//--------------------------------------------------------------------

    NumCards = ad2100_CountCards();                             printf("AD2100_Test
NumCards Opened = %d\n",NumCards);
    if (NumCards > 0) ad2100_error_code = ad2100_Open (1);    printf("AD2100_Test
ad2100_error_code = %d\n",ad2100_error_code);

    if (ad2100_error_code > 3) return(0);

            // ad2100_error_code:  0 = ok,
            //                     3 = ok, card already open
            //                     6 = Number of cards exceeded
            //                    13 = Kernel Driver not working
```

```
//---------------------------------------------------------------------
// 3)  SETUP CARD
//---------------------------------------------------------------------

    ad2100_Reset(1);                // Resets card's registers, Re-Arms Trigger
                                    // ==> Must always call before
ad2100_Digitize(1)

    ad2100_SetClockRate(1,1);       // Set clock to 50MHz
    ad2100_SetSegment_Size(1,0);    // Set segment size to 128
    ad2100_SetNumSegments(1,1);     // Set number of concatenated segments to 1
    ad2100_SetTrigPol(1,0);         // Set trigger input to positive edge
triggered



//---------------------------------------------------------------------
// 4)  DIGITIZE / WAIT FOR DONE / TRANSFER DATA TO "TempArray"
//---------------------------------------------------------------------

    ad2100_Digitize(1);             // Digitizes when TTL trigger is present (or
use "ad2100_SoftTrig()")
    ad2100_SoftTrig(1);

    for (DWORD x=0; x < 4000; x++) {
       if (ad2100_Digitize_Done(1) == 1) {          // Poll until Digitize = 1
           ad2100_TransferData(1,0,1,TempArray);    // Transfer data to User Array
           printf("TRANSFER DATA COMPLETE \n");
           break;
       }
    }

    for (DWORD x=0; x < 40; x++) printf("CHAN_1[%2d] = %5d   CHAN_2[%2d] =
%5d\n",x,(TempArray[x] & 0x0000FFF),x,((TempArray[x] >> 12) & 0x0000FFF) );



 //---------------------------------------------------------------------
// 5)  CLOSE DRIVER IF EXITING FROM MAIN PROGRAM
//---------------------------------------------------------------------

   ad2100_Close(1);   // Run this before quitting/shutting down master program,
                      // otherwise you can leave card open.

    return (0);
}
```

5 MISCELLANEOUS

5.1 Calibration

The AD2100 has no user feature to calibrate for input and output gain accuracy or offsets. The gains and offsets are calibrated at the factory to be within 3% at 25ºC and nominal voltages with inputs terminated by 50 ohms source.

5.2 Maintenance

No maintenance is required. However, a yearly calibration is recommended if the user desires to maintain the AD2100 modules' specified accuracy. Call factory for maintenance and/or extended warranty information.

Trademarks:
MS-DOS, Windows 3.1, Windows 95, Windows NT, Windows 2000, and Windows XP are registered trademarks of Microsoft Corporation.